CS 520

Theory and Practice of Software Engineering
Fall 2021

Collaborative development

October 19, 2021

Recap: Test driven development process

1. Add new test case(s) for a new feature and run all test cases
   - The new test case(s) should fail
2. Implement that new feature and run all test cases
   - All tests should pass
3. Refactor the implementation as needed to improve its quality and run all test cases
   - All tests should should still pass
4. Repeat

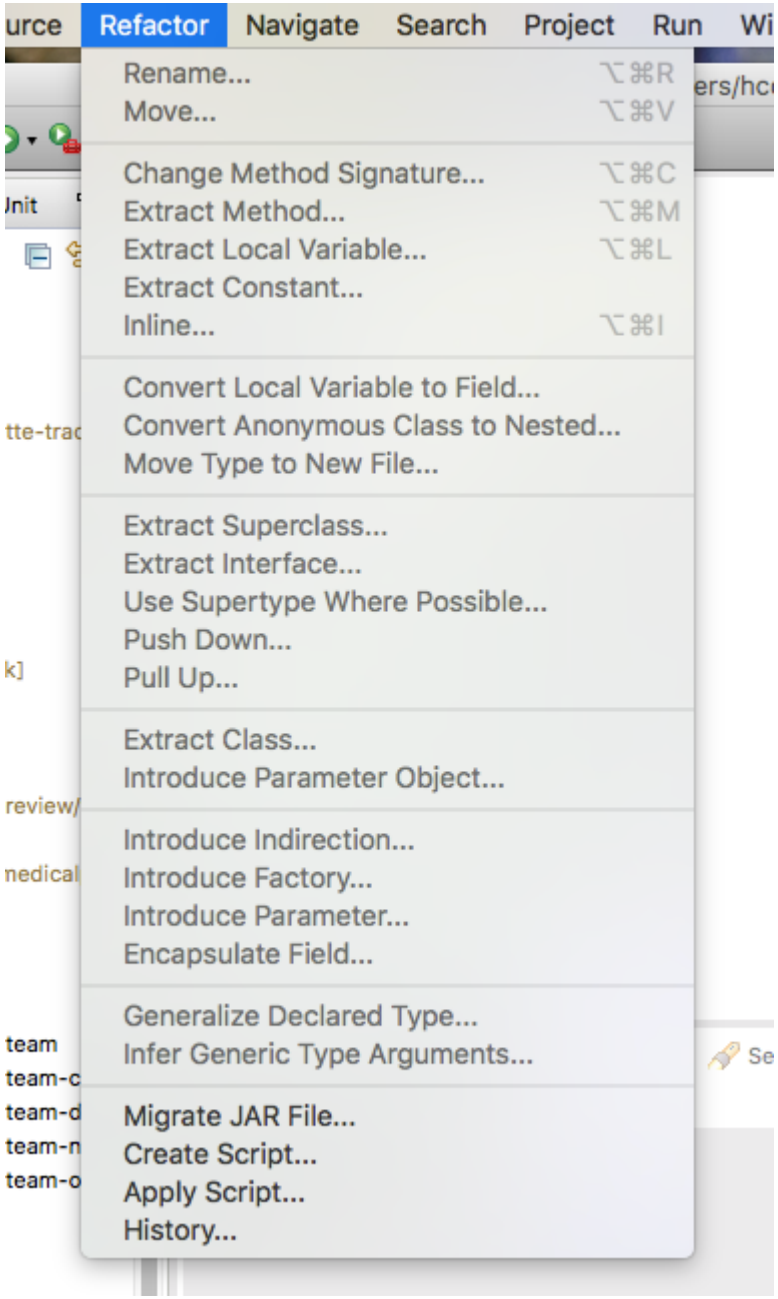
https://en.wikipedia.org/wiki/Test-driven_development

Recap: Some common refactoring patterns

- move class/field/method
- rename class/field/method
- extract class/field/method
- encapsulate field/method
- …

e.g., https://refactoring.com/catalog/

Recap: Automated support for those patterns



Today

- Agile development
- Scrum
- Pair programming
- Collaborative development exercise

Agile development

- Fast paced
- Frequent releases
- Developer centered
  - Do we need managers?

Scrum

- A very popular flavor of Agile to rapidly iterate in Sprints
  - Each Sprint develops then releases the product
- Three pillars:
  - Transparency
  - Inspection
  - Adaptation
- Used by large tech companies such as Facebook, Google, Microsoft

https://www.scrum.org

Three roles

- Product owner
  - represents the customer specifying the goal
- Development team
  - Performs Sprints
  - delivers software product that satisfies that goal
- Scrum master
  - buffer between team and outside world
  - prevents distractions, barriers

Many aspects of Scrum

- Sprints
- Stand-up meetings
  - What did I do yesterday?
  - What will I do today?
  - Do I see any impediment from our goal?
- Reviews

Pair programming

- Requirements specification, designing, implementing, testing, etc.
- Pair-work facilitates
  - transparency
  - no single point of failure
  - decision making
  - focus
  - creativity

Collaborative development exercise

- Further develop a Figure editor available here: https://github.com/LASER-UMASS/cs520-Spring2020.git

- Form pairs that will collaboratively work on:
  - specification
  - design
  - Implementation
  - testing

Figure editor

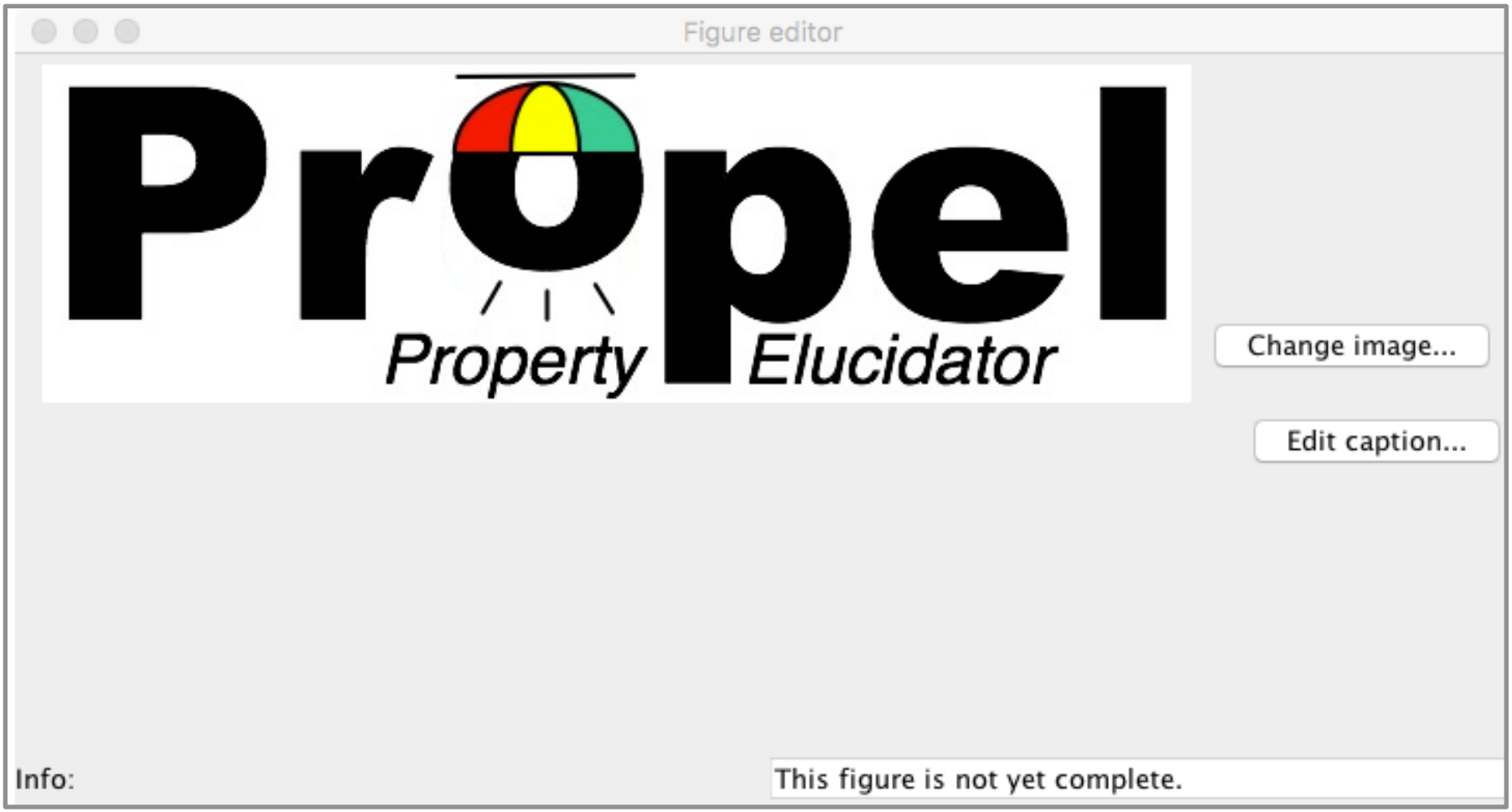
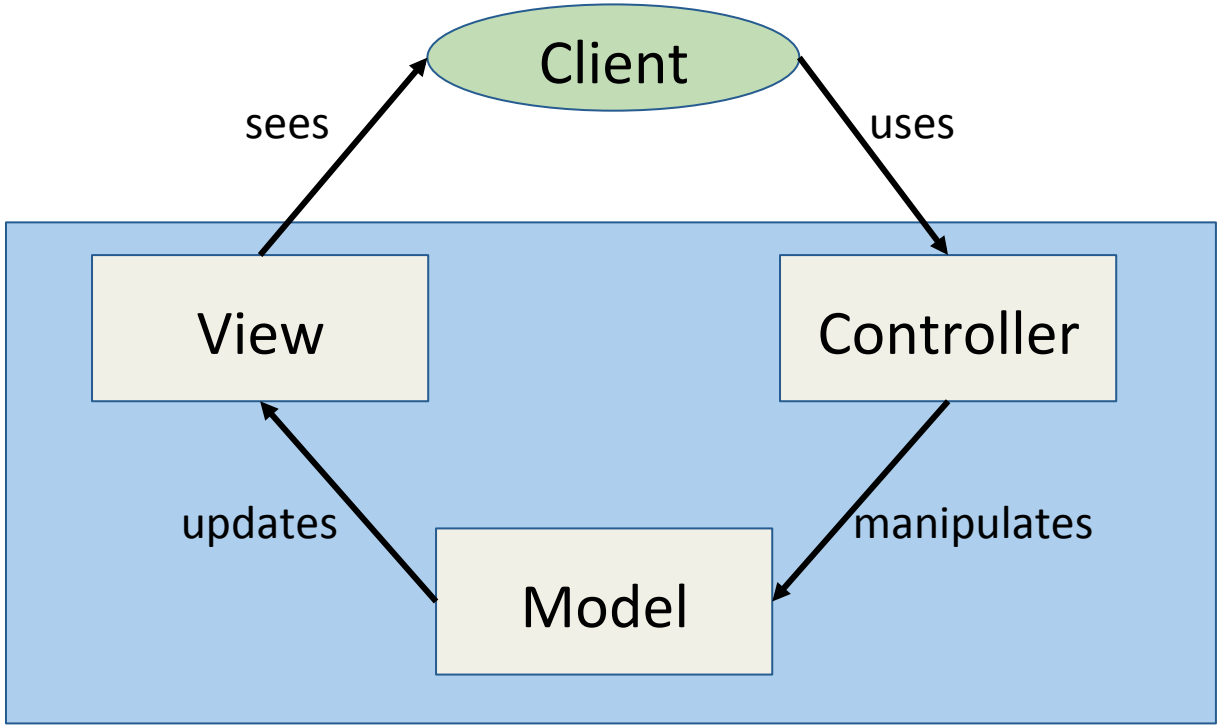


Figure editor (v1): MVC architecture



Separates data representation (Model),
visualization (View), and client interaction (Controller)

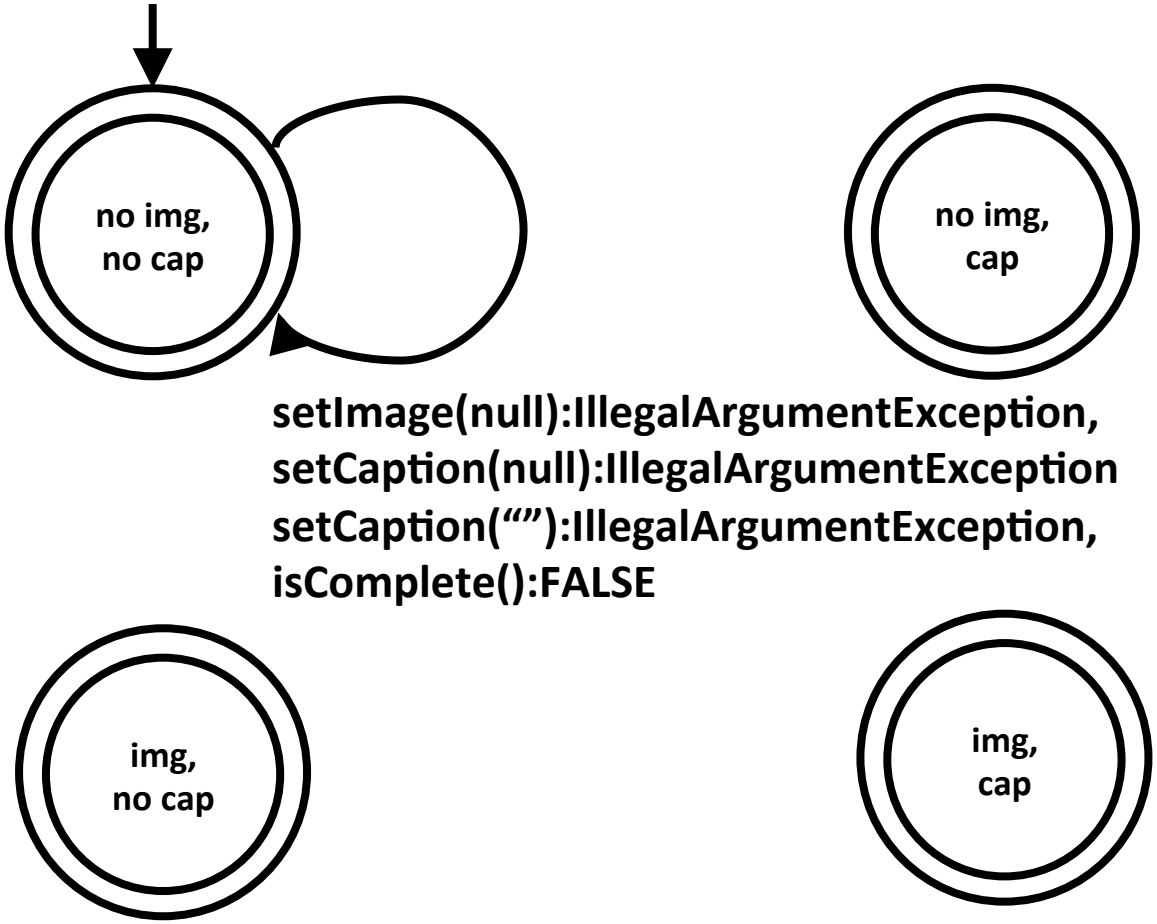
Figure editor (v1): Model API

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`java.lang.String`	**`getCaption()`**
`javax.swing.ImageIcon`	**`getImage()`**
`boolean`	**`isComplete()`** Returns true if this figure is complete, meaning its Image is non-null and its caption is non-null and non-empty, and false otherwise.
`void`	**`setCaption(java.lang.String newCaption)`** Sets the caption to the given non-null and non-empty String.
`void`	**`setImage(javax.swing.ImageIcon newImage)`** Sets the image to the given non-null ImageIcon.

Model (v1): FSA specification

- Complete the behavioral specification (written as an FSA)



NOTE) The violation state is implied (not shown).

Model (v1): Implementation

- Use the Model API and FSA for the implementation

setImage

```
public void setImage(javax.swing.ImageIcon newImage)
```

Sets the image to the given non-null ImageIcon.

Parameters:

`newImage` - The ImageIcon must be non-null

Throws:

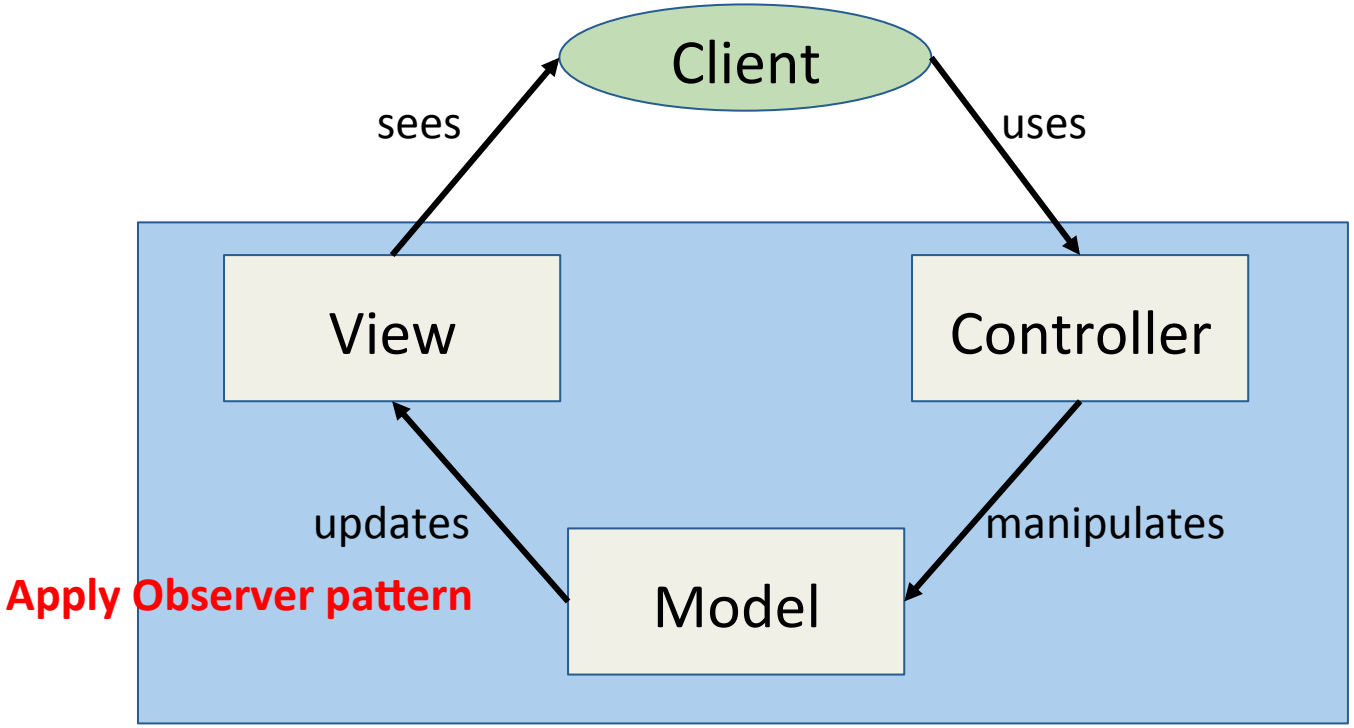
`java.lang.IllegalArgumentException` - if the ImageIcon is null

Figure editor (v1): Testing

- Write test cases for the Model

- All test cases should pass

Figure editor (v2): MVC architecture



Separates data representation (Model),
visualization (View), and client interaction (Controller)

Figure editor (v1): Observer pattern

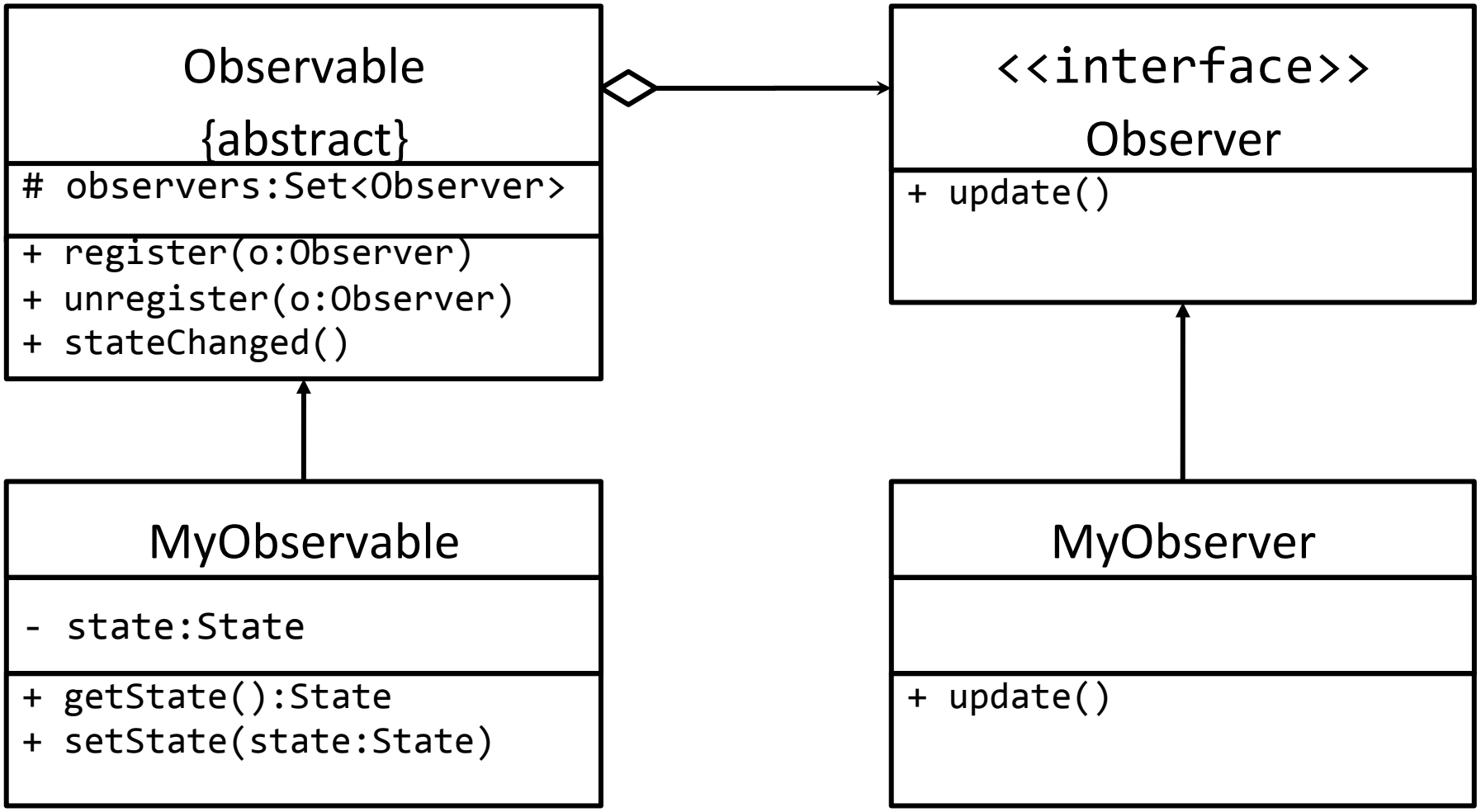


Figure editor (v1): Observer pattern

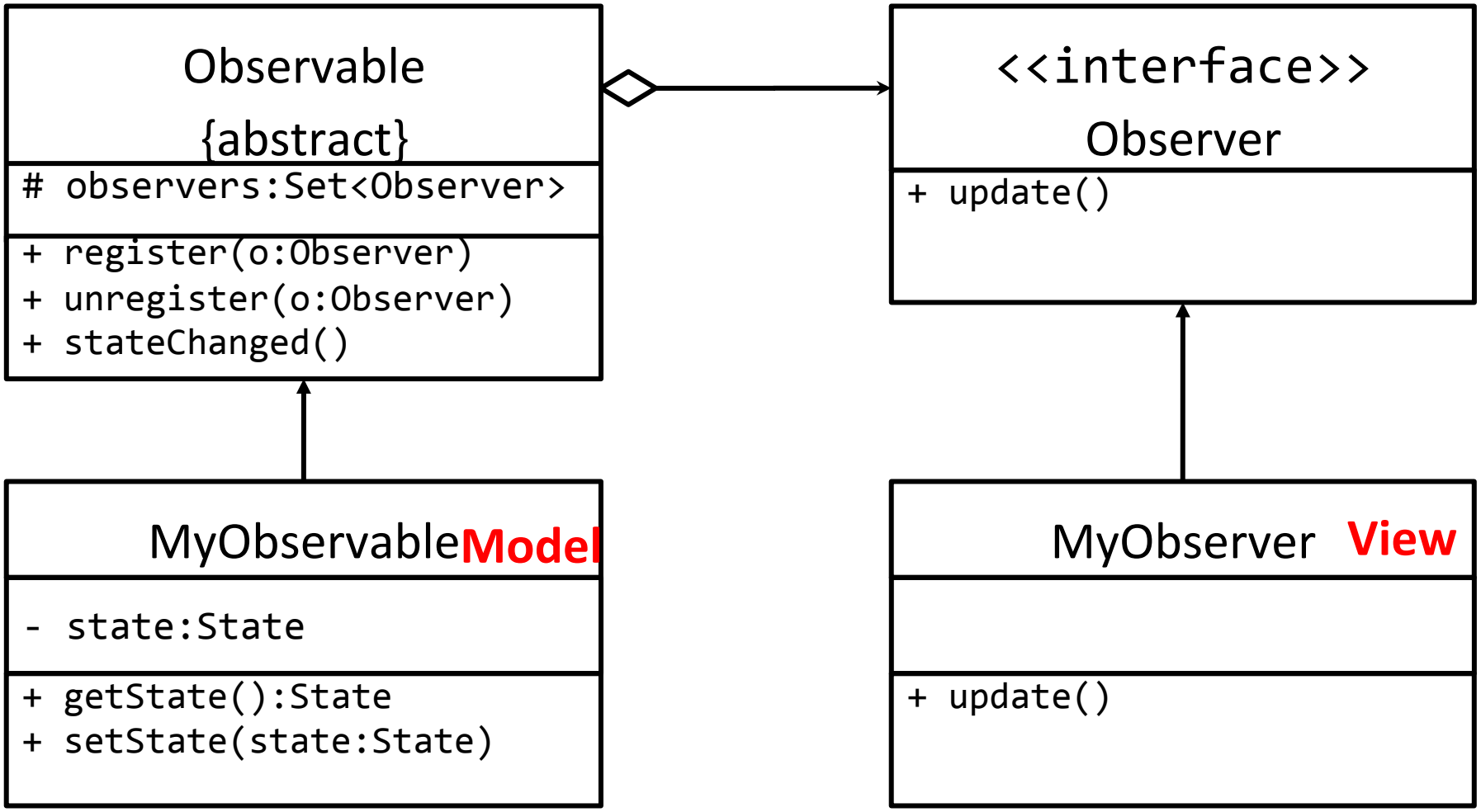


Figure editor (v2): Implementation

Java provides the following two classes:

- https://docs.oracle.com/en/java/javase/15/docs/api/java.desktop/java/beans/PropertyChangeSupport.html
- https://docs.oracle.com/en/java/javase/15/docs/api/java.desktop/java/beans/PropertyChangeListener.html

How could the Observer design pattern be implemented using these classes?

Figure editor (v2): Testing

- Perform regression testing with the existing test cases for the Model
- Add new test cases for the Observer design pattern

Topics covered

- Documentation, e.g.,
  - README, javadoc, internal comments
- Specification, e.g.,
  - Natural language, FSAs
- Architecture & design, e.g.,
  - Patterns (MVC, Observer)
  - Class diagrams
- Implementation
  - Pair programming
  - Java, Swing
- Testing
  - JUnit

Homework 2

- **Documentation:** Generate the javadoc
- **Design principles:** Implement 4 of the proposed fixes for violations of best practices
  - From the homework 1 solution
  - Your own
- **Design patterns:**
  - Briefly describe how to implement the Composite pattern in the view package
  - Identify the Observer pattern for the JButton class in Swing
- **Unit testing:** Implement additional test cases